Description

SOURCE INDEPENDENT FILE ATTRIBUTE TRACKING

Inventor: William E. Sobel

Technical Field

This invention pertains generally to computer file analysis, and more specifically to application independent gleaning of attributes concerning files in multiple formats.

Background Art

It is useful for computers connected to networks to scan incoming files and store associated file attributes. An attribute concerning a file can comprise any piece of information relevant to that file, such as its source, name, size or type. Stored file attributes can be useful both to the computer user and to various application programs. The user may want a record detailing files that were transmitted to a computer or that entered the computer's file system from external sources. Such information can also be utilized by various automated processes, such as a malicious computer code blocking system.

Computers connected to networks are vulnerable to network based malicious computer code attacks, such as worms, viruses and Trojan horses. As used herein, "malicious computer code" is any code that enters a computer without an authorized user's knowledge and/or without an authorized user's consent. Various blocking systems exist which attempt to block incoming malicious computer code. Information concerning past and present incoming files can be used by such systems to determine which files to block.

Some existing systems scan incoming files, and determine and store the name of the originating application (e.g., outlook.exe, iexplore.exe). However, such systems have no knowledge of the various file formats generated by different applications, and are unable to obtain further information about the files (e.g., the URL visited, an attachments sender's address).

What is needed are methods, computer readable media and systems that can glean and store file attributes concerning incoming files in a variety of formats, regardless of which applications generated the files.

Disclosure of Invention

The present invention comprises methods, computer readable media, and systems for gleaning file attributes independently of file format. A non-application specific file attribute manager (101) receives (201) a plurality of files (103) in a plurality of formats. The file attribute manager (101) scans (203) the plurality of received files (103), and gleans (205) attributes concerning each of the plurality of scanned files (103). The file attribute manager (101) stores (207) gleaned attributes concerning each of the plurality of scanned files (103) as records (105) in a database (107). The file attribute manager (101) indexes (209) the records (105) according to the contents of their associated files (103).

The features and advantages described in this disclosure and in the following detailed description are not all-inclusive, and particularly, many additional features and advantages will be apparent to one of ordinary skill in the relevant art in view of the drawings, specification, and claims hereof. Moreover, it should be noted that the language used in the specification has been principally selected for readability and instructional purposes, and may not have been selected to delineate or circumscribe the inventive subject matter, resort to the

claims being necessary to determine such inventive subject matter.

Brief Description of the Drawings

Figure 1 is a block diagram illustrating a high level overview of a system for practicing some embodiments of the present invention.

Figure 2 is a flowchart illustrating steps for performing some embodiments of the present invention.

Figure 3 is a flowchart illustrating steps for processing the receipt of multiple copies of the same file, according to some embodiments of the present invention.

Figure 4 is a flowchart illustrating steps for automatically deleting old records from the database, according to some embodiments of the present invention.

Figure 5 is a flowchart illustrating steps for a behavior blocking system to utilize gleaned file attributes according to some embodiments of the present invention.

The Figures depict embodiments of the present invention for purposes of illustration only. One skilled in the art will readily recognize from the following discussion that alternative embodiments of the structures and methods illustrated herein may be employed without

departing from the principles of the invention described herein.

Detailed Description of the Preferred Embodiments

Figure 1 illustrates a high level overview of a system 100 for performing some embodiments of the present invention. A non-application specific file attribute manager 101 receives incoming files 103 in a plurality of formats. The incoming files 103 can be received, for example, from a network connection or an external medium, such as a CD-ROM. The incoming files 103 can be generated by a wide variety of different application programs (e.g., word processors, spreadsheet programs, HTML editors, compilers, etc.), and thus can be in a wide variety of different file formats.

The non-application specific file attribute manager 101 scans the incoming files 103, regardless of their format. It is to be understood that this scanning can be done in conjunction with an additional function, such as anti-virus scanning of the incoming files 103, or as an independent process. Either way, the file attribute manager 101 gleans attributes concerning each of the plurality of scanned files 103 in the plurality of formats. Attributes can comprise any information concerning the

file, such as its size, its source, its name, etc. In some embodiments, the specific attributes to glean concerning a specific file 103 are a function of the protocol according to which the file 103 was transmitted. For example, a file 103 could be received according to Simple Mail Transfer Protocol (e.g., an incoming e-mail message). In that case, it could be appropriate to glean attributes such as the sender's address, the subject line and the digital signature status, as well as more universal attributes such as size and file name. In other embodiments, the specific attributes to glean concerning a specific file 103 are a function of the file 103 format. It is to be understood that the examples of attributes to glean as well as the associated gleaning criteria discussed herein are not all inclusive. Other examples will be readily apparent to those of ordinary skill in the relevant art in light of this specification. Which attributes to glean concerning which files 103 is a design choice.

The file attribute manager 101 stores gleaned attributes as records 105 in a database 107, such that a record 105 is created for each receipt of a file 103 of interest. Each record 105 stores at least some of the gleaned attributes. In some embodiments, the file attribute manager 101 stores all of the gleaned attributes,

and in other embodiments the file attribute manager stores various subsets of the gleaned attributes, as desired. Which attributes concerning which files 103 to store is a design choice.

The file attribute manager 101 indexes the attributes being stored as records 105 in the database 107 according to the contents of their associated files 103. In one embodiment, an index 109 is based on a secure hash of the associated file 103. In another embodiment, indexes 109 are based on cyclical redundancy checks of the associated files 103. Of course, other techniques for creating indexes 109 based on file contents are possible, and all such techniques are within the scope of the present invention. In any case, the indexes 109 can be subsequently used to retrieve stored database records 105 concerning files 103 for desired processing, for example by a blocking system.

As illustrated in Figure 1, the file attribute manager 101 can receive multiple copies of the same file 103. In Figure 1, the file attribute manager 101 receives two copies of File 2. When the file attribute manager 101 receives multiple copies of the same file 103, the file attribute manager 101 stores a separate database record 105 for each received copy, each record being indexed according

to the contents of the file 103. That way, each record 105 concerning the file 103 can be accessed by the single index 109. Later, a blocking system or other program analyzing received files can retrieve all available information on each copy of the received file 103 via the single index 109. This can be important, because the different copies of the received file 103 can have different attributes, for example because they were received from different sources. It is desirable to be able to determine that the separate records 105 map to different copies of the same file 103, so as to be able to perform a complete analysis thereon. In Figure 1, attributes concerning the two copies of File 2 are stored as Record 2A and Record 2B, both of which are pointed to by Index 2.

It is to be understood that although the non-application specific file attribute manager 101 is illustrated as a single entity, as the term is used herein a non-application specific file attribute manager 101 refers to a collection of functionalities which can be implemented as software, hardware, firmware or any combination of the three. Where a non-application specific file attribute manager 101 is implemented as software, it can be implemented as a standalone program, but can also be implemented in other ways, for example as part of a larger

program, as a plurality of separate programs, or as one or more statically or dynamically linked libraries.

In some embodiments the non-application specific file attribute manager 101 is incorporated into a server computer. In other embodiments, the non-application specific file attribute manager 101 is incorporated into a gateway or a client computer. In yet other embodiments, the non-application specific file attribute manager 101 is incorporated into other components as desired, for example a firewall, an intrusion detection system, an intrusion detection system application proxy, a router, one or more switch(es) and/or a standalone proxy. In some embodiments, the non-application specific file attribute manager 101 is distributed between or among more than one of the above and/or other components.

Figure 2 illustrates steps for performing some embodiments of the present invention. As described above in conjunction with Figure 1, the non-application specific file attribute manager 101 receives 201 a plurality of files 103 in a plurality of formats. The file attribute manager 101 scans 203 the plurality of received files 103, and gleans 205 attributes concerning each of the plurality of scanned files 103. As discussed above, the file attribute manager 101 stores 207 at least some gleaned

attributes concerning each of the plurality of scanned files 103 as records 105 in a database 107, indexing 209 the records 105 according to the contents of their associated files 103.

Figure 3 illustrates steps for processing the receipt of multiple copies of the same file 103, according to some embodiments of the present invention. The file attribute manager 101 receives 301 a plurality of copies of the same file 103. As described above, the file attribute manager stores 303 a separate record 105 for each received copy of the file 103, each record 105 being indexed 209 according to the contents of the file 103, such that each record 105 can be accessed by the single index 109.

In some embodiments of the present invention, the file attribute manager 101 automatically deletes old records 105 from the database 107, ensuring that the database 107 is kept current and free of obsolete records 105. Figure 4 illustrates steps for automatically deleting old records 105 from the database 107, according to some embodiments of the present invention. As discussed in conjunction with Figure 2, the non-application specific file attribute manager 101 receives 201 a plurality of files 103 in a plurality of formats. The file attribute manager 101 scans 203 the plurality of received files 103, and gleans 205

attributes concerning each of the plurality of scanned files 103. The file attribute manager 101 then stores 207 at least some gleaned attributes concerning each of the plurality of scanned files 103 as records 105 in a database 107. To keep the database 107 current, the file attribute manager 101 deletes 401 records 105 from the database 107 after the records 105 have been stored for a specific period of time. The specific period of time for which to store records 105 before deleting 401 them is a design choice.

Figure 5 illustrates steps for a behavior blocking system to utilize gleaned file 103 attributes according to some embodiments of the present invention. The blocking system examines 501 a file 103, which has already been scanned 203 as described above. In order to determine whether to block the incoming file 103 (e.g., from entering the computer, from executing, from performing certain functions while executing, etc.), the blocking system utilizes the index 109 based on the contents of the file 103 in order to retrieve 503 the associated record(s) 105 in the database 107. The blocking system proceeds to analyze 505 the attributes concerning the file 103 retrieved from the stored record(s) 105, and determines 507 a status of the file. This status can be used to determine

how to process the file 103. In some embodiments, the blocking system determines 507 that the file is legitimate and does not block 509 the file 103 (e.g., the blocking system allows the file 103 to enter the computer, or to execute, or to perform some other function). In other embodiments, the blocking system determines 507 that the file 103 is malicious, and blocks 511 the file 103 as appropriate.

As an example, the system 100 could first receive 201 an e-mail attachment "badfile.exe," which is known by name to contain malicious code. During the processing of the file 103 as described above in conjunction with Figure 2, relevant attributes concerning the file 103 will be gleaned 205 and stored 207 in a database 107 record 105, indexed 209 according to the contents of the file 103. The blocking system would then block 511 the file 103 from entering the computer, because of its known malicious status.

Later, suppose the same malicious file 103 is transmitted to the computer from another source, under the name "goodfile.exe." Because of the renaming of the file, the blocking system will not be able to identify it as being malicious based on its name alone. However, the system will scan 203 the file 103, and glean 205 and store

207 relevant attributes. When the blocking system receives 501 the malicious file 103, it will use the index 109 based on the file 103 contents to retrieve 503 the associated records 105 in the database 107. By analyzing 505 the file 103 attributes in the retrieved records 105, the blocking system can determine 507 that the received "goodfile.exe" is actually the same file 103 as "badfile.exe," a known malicious file 103. Accordingly, the blocking system will block 511 "goodfile.exe" from entering the computer.

In some embodiments, rules can be written, specifying to use gleaned file 103 attributes to process files 103 in specific ways. For example, a rule could specify to always allow executable files 103 attached to signed e-mails from trusted sources to execute without restriction. As explained above, the same file 103 can be received via multiple sources (or from the same source via multiple channels). When this occurs, multiple records 105 are stored 207 in the database 107 accordingly. The rule system can determine which rule(s) to apply 513 (most restrictive, least restrictive, etc.) when multiple records 105 exist. The specific rules to apply 513 and the specific manner in which to apply 513 them are variable design choices.

Of course, these are only examples of how a blocking system can use gleaned file 103 attributes in determining 507 which files 103 to block 511. Other examples will be readily apparent to those of ordinary skill in the relevant art in light of this specification. It will also be readily apparent to those of ordinary skill in the relevant art in light of this specification that a blocking system is only one type of system that can utilize file 103 attributes gleaned according to the present invention. of course, such gleaned attributes can be used by any type of system for any type of file analysis, as desired.

It will be understood by those of ordinary skill in the relevant art in light of this specification that the present invention enables non-application specific gleaning and storing of file attributes, such that the stored file attributes can later by utilized for analysis, for example by a blocking system.

As will be understood by those familiar with the art, the invention may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. Likewise, the particular naming and division of the modules, managers, features, attributes, methodologies and other aspects are not mandatory or significant, and the mechanisms that implement

the invention or its features may have different names, divisions and/or formats. Furthermore, as will be apparent to one of ordinary skill in the relevant art, the modules, managers, features, attributes, methodologies and other aspects of the invention can be implemented as software, hardware, firmware or any combination of the three. Of course, wherever a component of the present invention is implemented as software, the component can be implemented as a script, as a standalone program, as part of a larger program, as a plurality of separate scripts and/or programs, as a statically or dynamically linked library, as a kernel loadable module, as a device driver, and/or in every and any other way known now or in the future to those of skill in the art of computer programming. Additionally, the present invention is in no way limited to implementation in any specific programming language, or for any specific operating system or environment. Accordingly, the disclosure of the present invention is intended to be illustrative, but not limiting, of the scope of the invention, which is set forth in the following claims.

What is claimed is: